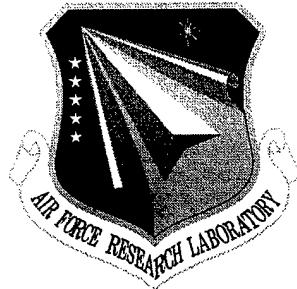


AFRL-IF-RS-TR-1999-136
Final Technical Report
June 1999



REAL-TIME PARALLEL BENCHMARKS

Honeywell, Inc.

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. D359

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

**AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK**

~~100% QUALITY INSPECTED~~

19990802 139

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-1999-136 has been reviewed and is approved for publication.

APPROVED:


RALPH KOHLER
Project Engineer

FOR THE DIRECTOR:


NORTHRUP FOWLER III, Technical Advisor
Information Technology Division
Information Directorate

If your address has changed or if you wish to be removed from the Air Force Research Laboratory Rome Research Site mailing list, or if the addressee is no longer employed by your organization, please notify AFRL/IFTC, 26 Electronic Pky, Rome, NY 13441-4514. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REAL-TIME PARALLEL BENCHMARKS

Brian VanVoorst
Rakesh Jha
Subbu Ponnuswamy
Chirag Nanvati
Luiz Pires

Contractor: Honeywell, Inc.
Contract Number: F30602-94-C-0084
Effective Date of Contract: 31 August 1994
Contract Expiration Date: 31 July 1998
Short Title of Work: Real-Time Parallel Benchmarks
Period of Work Covered: Aug 94 - Jul 98

Principal Investigator: Brian VanVoorst
Phone: (612) 951-7190
AFRL Project Engineer: Ralph Kohler
Phone: (315) 330-2016

Authorized for public release; distribution unlimited.

This research was supported by the Defense Advanced Research Projects Agency of the Department of Defense and was monitored by Ralph Kohler, AFRL/IFTC, 26 Electronic Pky, Rome, NY.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
<p>Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.</p>			
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED	
	Jun 99	Final 31 Aug 94 - 31 Jul 98	
4. TITLE AND SUBTITLE		5. FUNDING NUMBERS	
REAL-TIME PARALLEL BENCHMARKS		C - F30602-94-C-0084 PE - 62301E PR - 5581 TA - 20 WU - 78	
6. AUTHOR(S)		8. PERFORMING ORGANIZATION REPORT NUMBER	
Brian VanVoorst, Rakesh Jha, Subbu Ponnuswamy, Chirag Nanvati and Luiz Pires			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)		10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
Honeywell Inc. Honeywell Technology Center 3660 Technology Drive Minneapolis MN 55418		AFRL-IF-RS-TR-1999-136	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		11. SUPPLEMENTARY NOTES	
AFRL/IFTC 26 Electronic Pky Rome NY 13441-4514		AFRL Project Engineer: Ralph Kohler, IFTC, 315-330-2016	
12a. DISTRIBUTION AVAILABILITY STATEMENT		12b. DISTRIBUTION CODE	
Approved for public release; distribution unlimited.			
13. ABSTRACT (Maximum 200 words)			
<p>Benchmarks are used to evaluate and compare the performance of systems, using well-defined specifications and metrics. The Real-Time Parallel Benchmark (RTPB) suite is a collection of benchmarks designed to help real-time C3I parallel application designers evaluate parallel computing systems for real-time performance. Real-time C3I applications impose unique requirements on a computing platform, and at the time this project was conceived, existing parallel benchmark suites (even those focusing on C3I applications) did not adequately address these needs. The results obtained from this benchmark suite should help system architects, integrators and software designers better understand the factors that affect C3I real-time applications on parallel systems. Two goals of this project are: 1) to help reduce the cost and time of development cycles, and 2) to facilitate ongoing discussions between the C3I community and parallel system architectures about important issues that need to be addressed.</p>			
14. SUBJECT TERMS		15. NUMBER OF PAGES	
Benchmarks, High Performance Computing, Real-Time Computation, C3I, Parallel Processing		20	
16. PRICE CODE			
17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

1 Introduction

Benchmarks are used to evaluate and compare the performance of systems, using well-defined specifications and metrics. The Real Time Parallel Benchmark (RTPB) suite is a collection of benchmarks designed to help real-time C3I parallel application designers evaluate parallel computing systems for real-time performance. Real time C3I applications impose unique requirements on a computing platform, and at the time this project was conceived, existing parallel benchmark suites (even those focusing on C3I applications) did not adequately address these needs. The results obtained from this benchmark suite should help system architects, integrators and software designers better understand the factors that affect C3I real-time applications on parallel systems. Two goals of this project are: 1) to help reduce the cost and time of development cycles, and 2) to facilitate ongoing discussions between the C3I community and parallel system architectures about important issues that need to be addressed.

Benchmarks suites come in many forms. The RTPB suite is a "pen and paper" benchmark suite, meaning that the benchmarks themselves are *specifications*. The specification describes (in great detail sometimes) exactly what the benchmark is: what computations are to be performed, what input should be accepted and what the expected output should be, as well as a several other lower level details. In a sense, the specification states the objective of the benchmark and provides a detailed set of rules. When someone implements a benchmark based on the rules from the specification, then they have created a *benchmark implementation*, which is one realization or instance of the benchmark. Pen and paper benchmarks are used when there are considerable differences in the architectures on which that the benchmark may be implemented. By using the pen and paper approach, we let implementors choose the best possible implementation approach for a target platform. This is important, as we feel it is unfair to mandate one implementation (e.g. MPI) to be run on all machines, where for a given machine (e.g. a shared memory machine) a different implementation approach could produce better results. A side effect of pen and paper benchmarks is that they require considerable effort, both to develop a specification and to create an implementation.

Throughout the development of this benchmark suite we consulted many individuals including C3I domain experts and other benchmark suite developers to make these benchmarks as realistic and meaningful as possible. We worked with other researchers, end users and vendors to understand how our benchmarks would be used when they were completed. We spoke with the staff of high performance computing centers and directors at the High Performance Computing Modernization Program to explain the unique requirements and resources required by real-time C3I applications. This final report summarizes our observations and provides some suggestions for the future, both in terms of additional research and benchmark maintenance.

The rest of this report is organized as follows: Section 2 describes the importance of the RTPB effort, Section 3 presents a brief account of our accomplishments, Section 4 describes what we learned, Section 5 discusses the technology transfer that happened before completing this project, Section 6 offers suggestions for future work, and Section 7 presents our conclusions.

2 Program Goals

The amount of data to be processed by a C3I system is increasing rapidly as the quantity of the available information increases, and the expected quality of the results increases. Furthermore, real-time deadlines to process information are becoming shorter, as response times need to increase. Next generation sensors will produce terabytes of information that must be processed in real-time. To complicate matters, cost constraints force system designers to adopt commodity COTS components as the building blocks for their systems instead of special purpose hardware. The combination of these factors requires the use of commercial parallel processing systems for real-time C3I applications.

A system designer must consider several factors when choosing hardware for an embedded C3I application. Some of the relevant factors that must be considered include: single processor speed, scalability, real-time support and performance, efficiency of the algorithm on the machine, problem mapping, communication and memory requirements, fault tolerance, reliability, size, weight, power and the list goes on. Even if there are just a handful of architectures to consider, the amount of trade-off analysis required is staggering. Because of the amount of work involved, it is usually only the easily available metrics that are used for the comparison (size, weight, power, processor speed in Mhz, published communication latency). The most important factor, the interactions between the actual algorithm and the system under study, are rarely considered. The Real Time Parallel Benchmark Suite provides efficient mechanisms to make these comparisons easily.

The benchmarks provide a new metric for comparison that allows systems to be evaluated side-by-side in an "apples-to-apples" manner. This new metric captures the real world interactions between applications and hardware with real input data, and reflects the kind of performance that will be seen when the system is deployed.

Suppose a system architect has a C3I application with intensive computational requirements and real-time deadlines. From the system architects point of view, the best possible situation is that there already exist benchmark results that capture the main computational and real-time requirements of the application. If the results are not available, but a similar benchmark exists, the system architect can port and run the benchmark on the platform of interest. If there is no such benchmark available, the architect can follow the methodology we provide to specify a new benchmark and ask that vendors run this benchmark and submit results for comparison. Either way, in short order a fair comparison can be made between the platforms under consideration.

A secondary benefit is that the benchmarks serve to increase awareness and interest in real-time C3I applications on parallel machines. The NAS parallel benchmarks, which have become a standard for comparing machines for scientific computing, serve as an example. Because of the popularity of the NAS benchmarks and the emphasis placed on their results at parallel computing conferences, vendors compete to improve their performance for CFD type applications. If the Real Time Parallel Benchmarks can foster the same kind of interest, the DoD will see the benefit in the form of increased attention paid to their applications by researchers and hardware designers.

A third benefit of the benchmarks is that it provides the DARPA research community with a wealth of freely available, unclassified example applications. Several researchers who work with the technology DoD uses (FPGAs, compilers, communication protocols, ...) may not necessarily have a background in C3I. Because of this, some researchers have a hard time showing the relevance of their work in the domain in which the DoD would use it. The benchmarks have become a resource university researchers are drawing upon to create demonstrations that are relevant to their program managers. Several examples of this appear in our section on technology transfer.

3 Scope and Accomplishments

In the end we created ten benchmarks - two application level benchmarks (Adaptive beamforming and Multiple hypothesis testing), three kernel level benchmarks, (associative gating, constant false alarm rate, and matched filter) and five low level benchmarks (three clock benchmarks and two communication benchmarks). The kernel and application level benchmarks represent topics of significant interest to the DoD, namely image processing, image understanding, signal processing, filtering, pattern recognition, and classification problems. The low level benchmarks are of interest to anyone who is implementing a real-time algorithm on a parallel machine.

Work accomplished under this program, is listed here in Figure 1, with cross references to detailed reports.

1. A document describing the rational and criteria for benchmarks selection[8].
2. Creation of five low level benchmark specifications [1]. gclock, rtcomm1, rtcomm2, lclock and lsclock.
3. Creation of three kernel level benchmark specifications [3]. CFAR, Associative Gating and Matched Filter.
4. Creation of two application level benchmark specifications [5], Multiple Hypothesis Testing, and Adaptive Beamforming.
5. Creation of a benchmark real-time parallel benchmarking methodology [7].
6. Creation of a portable and reusable benchmarking harness to facilitate the real-time benchmarking of parallel machines efficiently.
7. Creation of data sets, acceptance tests, and results for each of the 10 benchmarks.
8. Creation of a portable sample implementation for each of the five low level benchmarks
9. Creation of a portable sample implementation for each of the three kernel level benchmarks.
10. Creation of a portable sample implementation for each of the two application level benchmarks.
11. Performance results for each of the low level sample implementations for at least two different machines [2].
12. Performance results for each of the kernel level sample implementations for at least two different machines [4].
13. Performance results for each of the application level sample implementations for at least two different machines [6].
14. Written description of all implementations and an analysis of the results [2,4,6].
15. A paper presented at SIAM's Parallel Processing for Scientific Computing 1997 conference detailing our work [10].
16. An analysis of what is necessary to make a third application level benchmark based on Automatic Target Recognition.
17. Presentation materials from several PI meetings and program reviews.
18. A final report describing the highlights of this work (this document).

4 What was learned from this program

We have divided what we learned into two sub-sections. What we learned about benchmarking real-time parallel machines is discussed first. This section discusses benchmark specifications, gaining access to platforms, and the end users of the benchmarks. The second section describes what we learned from running the benchmarks themselves, and our conclusions from studying the results we collected.

4.1 Lessons from developing and running real-time benchmarks on parallel machines

Benchmarking real-time parallel machines comes with its own set of challenges, which are outlined in our benchmarking methodology document [7]. The methodology describes all of the benchmarking issues that must be addressed. In this section we describe some of the other lessons learned such as those related to current practices in parallel machines, how to develop benchmarks, the community's reaction to the benchmarks, and alternative uses for the benchmarks.

4.1.1 Pen and Paper benchmarks are expensive to develop and implement

Pen and paper benchmarks are very flexible as they allow benchmark implementors to choose the best methods to implement the benchmark onto a specific machine. However, this flexibility comes at a cost. Benchmark specifications are expensive to develop, and new implementations are expensive to create. As standard and portable tools evolve (such as MPI, libraries, and other API's) for embedded systems, it may be possible to create "code" benchmarks, where the benchmarker need only run a standard piece of source code on any platform to gather results.

4.1.2 Selection of real-time parallel platforms to benchmark is limited

Most high performance computing centers provide access to platforms that are commercially available, and most of these centers' computers have a workload made up mostly of scientific applications (CFD, computational chemistry, statics and dynamics modeling). These applications are typically run in a batch mode from datasets stored on disk. Few of these machines have real-time operating systems installed (even though vendors may offer real-time environments, the majority of the users have no need for it). Similarly, most of the jobs on these machines are compute-bound, whereas C3I applications with real-time input can be input-bandwidth bound.

Unfortunately, the embedded systems vendors who do have parallel real time environments were often unable or unwilling to allow us free access to their machines for the purposes of benchmarking. We found that these vendors see benchmark suites as an expensive burden and they feel a catch-22: either they let others benchmark their machines (whom they do not trust to represent in the best light) or they do the benchmarking themselves (at a considerable cost). Therefore most vendors will not report benchmark results unless they are competing for a contract. The result is that sample implementations get run on publicly accessible machines that may not have a real-time environment. In the end vendors may produce results for their machines, or otherwise independent researchers with access to these platforms may collect results.

It is our recommendation that future benchmarking projects secure funding to pay for time on vendors machines, or establish partnerships in the program to ensure access to these platforms.

4.1.3 Benchmarks are useful even to those who do not want to do benchmarking

Several other researchers in the Embedded Systems program (and other DoD funded programs) have requested the benchmarks as sample applications to test their systems and subsystems. While this is an unexpected use of the benchmarks, it indicates that the benchmarks are fulfilling a need that was previously unnoticed, namely researchers need freely accessible DoD applications to demonstrate and test their work. This need may be explored further by DARPA and other agencies.

4.1.4 Reusable benchmark harness proved to be a useful tool

At the beginning of our work we created a portable benchmark harness. The harness provided many of the tools and services we needed for benchmarking platforms that were above and beyond what was required

of a benchmark itself. For example, the benchmark harness handled issues of global time, sending input to the benchmark, collecting results, timing the computations, checking results, and increasing the input rate. We re-used the harness on each of our application and kernel level benchmarks. Not only did it save us time, it provides each of our sample implementations with a common interface which should be beneficial to those who work with the sample implementation source code.

4.2 Lessons from the benchmarks

Detailed observations on the results of each of the benchmarks is described in their respective reports [2,4,6]. However some overall observations are presented here.

4.2.1 Most applications have considerable memory requirements

Many of the kernel and application level benchmarks require nearly 256 Megabytes of memory per node, and in some cases even more. In the Associative Gating benchmark [4] it was noted that 512 Megabytes or more would be required to make an efficient parallel implementation. This much memory was not available on the machines tested. We observe that many future C3I applications implemented on parallel machines may be memory constrained on typical machine configurations.

4.2.2 True shared memory implementations may be more efficient (but less portable)

In the early 90's the high performance computing marketplace produced parallel systems that were scalable to a massive number of processors (hence the name "massive parallel processing"). This resulted in machines such as the Intel iPSC/860 Hypercube, Intel Paragon, IBM SP2, Thinking Machines CM5, the Cray T3D and T3E to name a few. These distributed memory machines were built from commodity processors and custom high speed interconnects. These systems allowed for hundreds (and in some cases thousands) of processors to be connected to offer the fastest machines possible. To ensure portability of parallel applications between systems, the Message Passing Interface (MPI) standard was created. Shared memory machines did not get much attention due to the difficulty in making scalable memory backplanes.

By the mid 90's consumers recognized that vendors invested so much time building these massive systems that by the time they were deployed, the compute processors used in individual nodes were out of date and slow compared to high-end desktop machines. The next major shift (still underway) was to replace traditional parallel supercomputers with networks of cheap workstations whose processors could be upgraded easily. SGIs, Suns, Cray J90s, and even PCs were used as the individual compute nodes. ATM switches, FIDDI, HIPPI, Myrinet and other high performance switches were used as interconnect technologies. Designers put as much compute power in each node as possible. Shared memory machines, with as many as 16 processors, were used as individual nodes, several of which would be connected together and treated as one machine. With this design, data was moved through a heterogeneous network; some communication happened in shared memory, other communication went over a switch. To simplify the programming model, programmers continued to communicate using message passing (MPI), regardless of whether the communication was occurring via shared memory or a network connection.

Currently vendors such as SGI are building commercial systems with similar designs. The SGI Origin 2000 is a cluster of shared memory machines. The system can scale to hundreds of processors. However SGI supports one memory address space for the entire system, allowing the programmer to use shared memory constructs instead of MPI. Portability is still a problem because the constructs used for synchronization are not standardized across manufacturers. What is lacking is a common shared memory API similar to the message passing standard provided by MPI.

As noted in the benchmark result reports, in some cases a shared memory implementation can offer greater scalability and performance than a message passing implementation. Since our implementations were

sample implementations, portability was a primary concern. Since most systems support MPI, we chose to implement our benchmarks using MPI, giving tips to developers as to how shared memory may improve performance. As the combination of shared and distributed memory becomes more commonplace, we feel that new tools to support a hybrid parallelization approach will become standardized, and more efficient portable implementations can be built in the future.

4.2.3 Scalability may be limited by factors other than aggregate processing power

Many of our benchmarks increase the rate of input to stress the machines ability to meet deadlines and establish a breakdown point. When a breakdown point is found for a set of processors, the number of processors is increased and the experiment is performed again. If the benchmark implementation is scalable it is expected that the sustainable input rate would double as the number of processors is doubled. This conventional wisdom does not always hold true as it assumes the reason for a deadline miss is due to insufficient compute resources, and that adding more processors will allow for a shorter deadline. The speed at which input can be processed consistently without missing a deadline can be dependent on many other factors, including bisection bandwidth of the innerconnect and latency for communication. These resources must also scale as the number of processors increases. Furthermore, interruptions and delays introduced by the operating system services must continue to go down when using more and more processors to achieve perfect real-time scalability. The opposite is more likely to happen, as the OS must manage communication channels with more and more processors. The result is that a benchmark with a deadline may appear to be less scalable even though the computations scale well. This is an accurate reflection of how the system will perform or real applications when deadlines are present.

4.2.4 Acceptance tests for real-time behavior must be carefully stated

Acceptance tests are used to determine if a benchmark is meeting it's real-time deadlines. The test must take into account measurement errors, granularity of the clock, clock skew and make measurements on the appropriate processor. In some benchmarks the processor to measure is the one that takes the longest time, which may change over time. Our benchmarking methodology document [7] addresses this issue for the benchmarker.

4.2.5 Machines that run time synchronization protocols can cause bad measurements

We found that machines that run clock synchronization protocols such as NTP can cause bad measurements in benchmarks. NTP can cause clocks to "jump" to synchronize, which ruins the timing of a benchmark, and leads to untrustworthy results. Our benchmark harness has a test for this type of behavior, and if it is detected it is reported.

4.2.6 Different clocks on the same system can give dramatically different results

On a modern machine a programmer has access to many different functions that can be used for timing. Many however have considerable overhead. For example, one MPI implementation tested on a Mercury had a MPI_Wtime() call that gave considerably worse results than the Mercury's hardware clock. Similarly, we found that the SGI special hardware clock reading routines gave much better results than any other operating system level clock calls. It is advisable that benchmarks and programmers familiarize themselves with all the clock options on a platform and select the clock that gives them the best granularity possible with the least overhead.

5 Technology transfer success stories to date

Several corporations and universities have already requested the benchmarks for their use. Those who have

requested advanced access to the benchmarks before the project was completed include: Hughes, BBN Technologies, Stanford University, the University of Michigan, The University of Massachusetts, and Michigan Technological University. Some of these groups are using the benchmarks to collect results and compare platforms, while others are using them as test suites to exercise systems and tools. Other benchmarking efforts have adopted benchmarks from this suite, namely the Adaptive Computing Benchmark project under DARPA's Adaptive Computing Systems program.

The distribution plan [9] outlines our suggestions on how access to the benchmarks should be managed so that other organizations may benefit from them in the future.

6 Future research and benchmark maintenance work

We have identified the following additional tasks to benefit this effort.

6.1 Implementation of the distribution plan

We provided a straightforward distribution plan [9] that should provide a reasonable mechanism for others to benefit from the benchmarks. It is important that DARPA takes steps to make this work available soon so that it can be adopted and used by others. The technology transfer opportunities we have had already are an indication of the strong interest in this work.

6.2 Workshops on using the benchmarks

Workshops provide an excellent way for researchers, vendors, and system designers to become familiar with the benchmarks and exchange ideas. Providing a way for members of the community to quickly ramp-up on the benchmark suite is a necessary step to ensure its long-term success.

6.3 Collection of additional results

The results we have captured are useful for reference by others but only represent a snapshot in time. As machines, tools, algorithms and operating systems advance, these results will become out of date. It is important that new results are continually collected to chart the progress of the benchmark performance of new machines.

6.4 Benchmarks in new areas

We have identified three benchmark suites that if developed could have a similar benefit to the C3I and embedded computing community that this program has had. A short description of each idea follows.

6.4.1 Benchmarks for wide area computing over the NGI

DARPA is a major contributor to the overall research and development of the NGI. Collectively thousands of researchers are working to make the NGI a reality. We feel that a carefully constructed set of benchmarks will allow DARPA to gauge if 1) the individual technologies contributed to the NGI achieve the goals set for them, and 2) if the NGI provides the performance and services needed by the DoD applications.

6.4.2 Benchmarks for heterogeneous systems

Many embedded solutions do not use a homogenous computing or communication system, but instead are complex systems comprised of multiple CPU's, special purpose accelerator boards, FPGA's, buses, crossbars, and the like. Benchmarking a heterogeneous system requires a different methodology than

benchmarking traditional homogenous systems. Adaptations of the benchmark methodology already created under this contract, along with sample implementations, would allow for better understanding of how C3I applications interact with the components of heterogeneous systems.

6.4.3 Benchmarks for fault tolerance

Fault tolerance is a key component of nearly every mission critical DoD application. There are currently no benchmarks designed to measure this important requirement of applications and the platforms they run on. We feel strongly that if DARPA were to fund the development of a fault-tolerant benchmark suite it would have a very positive impact for the DoD.

7 Conclusions

We have developed a real-time parallel benchmark suite for DoD applications, a first of its kind. We have also developed a methodology that explains how we created, ran and collected results from the benchmarks. Our suite currently has 10 benchmarks: 2 application level [5], 3 kernel level [3] and 5 low-level [1]. We have created sample implementations of each one and have run them all on two or more platforms. The results are discussed in our benchmark reports [2,4,6].

The Real Time Parallel Benchmark suite fulfills a need and is complementary to the work performed at MITRE [11, 14], and the C3I Parallel Benchmark Suite [12, 13] developed at Honeywell. Together these three projects have produced over twenty-five benchmarks for the C3I community. These benchmarks have the potential to assist future system designers in making decisions about the hardware and software they choose. They can also be of benefit to other researchers who wish to use them for their own experimentation. To help foster an environment where these benchmarks will be adopted and utilized, we have prepared a distribution plan. If this plan is adopted, we feel that it will result in a growing interest in C3I applications and the systems that run them, which should have an overall benefit to the DoD community.

8 Acknowledgments

We would like to thank the following computing centers for providing their resources: NASA Ames Research Center, Arctic Region Supercomputer Center, Maui High Performance Computing Center, University of Minnesota, Mississippi State University.

We would also like to thank the following individuals for their help and guidance: Ralph Kohler at AFRL, Jose Munoz at DARPA, Robert Bernecke at NRAD, Benoit Champagne at the University of Quebec, Richard Games at MITRE, Craig Lund at Mercury Computer Systems, Glen Ladd and Rob Moore at Hughes Research Labs, and David Bailey at the NASA Ames Research Center.

Appendix A References

1. "Low-Level Benchmark Specifications", Technical Information Report (C006), Rome Laboratory, Contract Number F30602-94-C-0084, August 1998
2. "Low-Level Benchmark Results", Technical Information Report (C007), Rome Laboratory, Contract Number F30602-94-C-0084, August 1998
3. "Kernel Level Benchmark Specifications", Technical Information Report (C008), Rome Laboratory, Contract Number F30602-94-C-0084, August 1998
4. "Kernel-Level Benchmark Results", Technical Information Report (C009), Rome Laboratory, Contract Number F30602-94-C-0084, August 1998
5. "Application Level Benchmark Specifications", Technical Information Report (C0010), Rome Laboratory, Contract Number F30602-94-C-0084, August 1998
6. "Application Level Benchmark Results", Technical Information Report (C0011), Rome Laboratory, Contract Number F30602-94-C-0084, August 1998
7. "Benchmarking Methodology", Technical Information Report (C005), Rome Laboratory, Contract Number F30602-94-C-0084, August 1998
8. "Benchmark Requirements and Selection", Technical Information Report (C004), Rome Laboratory, Contract Number F30602-94-C-0084, August 1998
9. "Benchmark Distribution Plan Report", Technical Information Report (C012), Rome Laboratory, Contract Number F30602-94-C-0084, August 1998
10. "A Real-Time Parallel Benchmark Suite", B. VanVoorst, L. Pires, R. Jha, Proceedings of SIAM's Parallel Processing for Scientific Computing, SIAM, March 1997.
11. "Real-Time Embedded High Performance Computing: Application Benchmarks", C. Brown, M. Flanzbaum, R. Games, J. Ramsdell, MITRE Technical Report, MTR 94B0000145, October 1994.
12. "C3I Parallel Benchmark Suite Final Report", Technical Information Report (A012), Rome Laboratory, Contract Number F30602-94-C-0084, May 1998
13. "The C3I Parallel Benchmark Suite - Introduction and Preliminary Results", R. Metzger, B. VanVoorst, L. Pires, R. Jha, W. Au, M. Amin, D. Castanon, V. Kumar, Proceedings from Supercomputing 96, 1996.
14. "Benchmarking Methodology for Real-Time Embedded Scalable High Performance Computing", R. Games, MITRE Technical Report, MTR96B0000010, March 1996.

DISTRIBUTION LIST

addresses	number of copies
RALPH KOHLER AFRL/IFTC 26 ELECTRONIC PARKWAY ROME NY 13441-4105	3
HONEYWELL, INC. HONEYWELL TECHNOLOGY CENTER 3660 TECHNOLOGY DRIVE MINNEAPOLIS MN 55418	3
AFRL/IFOIL TECHNICAL LIBRARY 26 ELECTRONIC PKY ROME NY 13441-4514	1
ATTENTION: DTIC-OCC DEFENSE TECHNICAL INFO CENTER 8725 JOHN J. KINGMAN ROAD, STE 0944 FT. BELVOIR, VA 22060-6218	2
DEFENSE ADVANCED RESEARCH PROJECTS AGENCY 3701 NORTH FAIRFAX DRIVE ARLINGTON VA 22203-1714	1
ATTN: NAN PFRIMMER IIT RESEARCH INSTITUTE 201 MILL ST. ROME, NY 13440	1
AFIT ACADEMIC LIBRARY AFIT/LDR, 2950 P. STREET AREA B, BLDG 642 WRIGHT-PATTERSON AFB OH 45433-7765	1
ATTN: SMDC IM PL US ARMY SPACE & MISSILE DEF CMD P.O. BOX 1500 HUNTSVILLE AL 35807-3801	1

TECHNICAL LIBRARY D0274(PL-TS)
SPAWARSYSCEN
53560 HULL STREET
SAN DIEGO CA 92152-5001

1

COMMANDER, CODE 4TL0000
TECHNICAL LIBRARY, NAWC-WD
1 ADMINISTRATION CIRCLE
CHINA LAKE CA 93555-6100

1

CDR, US ARMY AVIATION & MISSILE CMD
REDSTONE SCIENTIFIC INFORMATION CTR
ATTN: AMSAM-RD-08-R, (DOCUMENTS)
REDSTONE ARSENAL AL 35898-5000

2

REPORT LIBRARY
MS P364
LOS ALAMOS NATIONAL LABORATORY
LOS ALAMOS NM 87545

1

AFIWC/MSY
102 HALL BLVD, STE 315
SAN ANTONIO TX 78243-7016

1

USAF/AIR FORCE RESEARCH LABORATORY
AFRL/VSOSAC(LIBRARY-BLDG 1103)
5 WRIGHT DRIVE
HANSCOM AFB MA 01731-3004

1

ATTN: EILEEN LADUKE/D460
MITRE CORPORATION
202 BURLINGTON RD
BEDFORD MA 01730

1

OUSD(CP)/DTSA/DUTD
ATTN: PATRICK G. SULLIVAN, JR.
400 ARMY NAVY DRIVE
SUITE 300
ARLINGTON VA 22202

1

***MISSION
OF
AFRL/INFORMATION DIRECTORATE (IF)***

The advancement and application of information systems science and technology for aerospace command and control and its transition to air, space, and ground systems to meet customer needs in the areas of Global Awareness, Dynamic Planning and Execution, and Global Information Exchange is the focus of this AFRL organization. The directorate's areas of investigation include a broad spectrum of information and fusion, communication, collaborative environment and modeling and simulation, defensive information warfare, and intelligent information systems technologies.